MATH/AMATH 516, Spring 2009 Homework 2 Due May 1 at 10:30 AM

You can discuss the problems with each other, but you must write up your answers on your own. Feel free to ask for a hint if you get stuck. [The page/equation numbers are from *Nonlinear Programming*, 2nd edition, 1999.]

The answers to the two * problems are to be turned in jointly with your (randomly chosen) partner.

#1. Suppose that $f : \Re^n \mapsto \Re$ is C^1 on $\Re^n$ and $\|\nabla f(x) - \nabla f(y)\| \le L\|x - y\|$ for all $x, y \in \Re^n$. Let $\{x^k\}$ be generated by the method $x^{k+1} = x^k + \alpha^k d^k$, with

$$g^{k'} d^k \le -c_1 \|g^k\|^2, \quad \|d^k\| \le c_2 \|g^k\| \qquad (0 < c_1 \le c_2)$$

and α^k given by Armijo rule (with parameters s, σ, β) whenever $g^k \neq 0$. Prove that

$$\alpha^k \ge \min\left\{ s, 2\frac{(1-\sigma)c_1\beta}{Lc_2^2} \right\}.$$

You can use the fact that $f(x+y) - f(x) \le \nabla f(x)'y + \frac{L}{2}\|y\|^2$ for all $x, y \in \Re^n$ (see Proposition A.24 of the 1999 edition). [Hint: Divide into two cases: (i) $\alpha^k = s$ and (ii) $\alpha^k < s$. In case (ii), α^k/β must violate the sufficient descent condition in the Armijo rule.]

#2.* [Dealing with Nondifferentiable Function.] Let $f : \Re^n \mapsto \Re$ be C^1 on $\Re^n$. In compressed sensing, we seek a minimim x^* of f that is sparse in the sense that x^* has few nonzero components. This can be formulated as the optimization problem

$$\min_{x \in \Re^n} f_c(x) := f(x) + c\|x\|_1,$$

where $\|x\|_1 = |x_1| + \cdots + |x_n|$ is the 1-norm and $c \ge 0$. Notice that $\|\cdot\|_1$ is a convex function, so f_c is a convex function whenever f is a convex function. However, f_c is not differentiable at points having some zero component, and a gradient descent method applied directly to f_c can get stuck at a non-stationary point.

a) One way people deal with the nonsmoothness of $|t|$ is to approximate it by a smooth convex function, say $\sqrt{t^2 + \mu}$, with $\mu > 0$ a small constant (say, $\mu = 10^{-6}$). What are the possible drawbacks? (Hint: Consider the Hessian of the approximate function. You might try this on the Rosenbrock or Powell example, with say, $c = 1$ or $c = .1$ and see what happens.)

b) A second way is to use the fact $|t| = \min\{r + s \mid r \ge 0, s \ge 0, r - s = t\}$ and reformulate the problem as

$$\min_{u,v \in \Re^n} \{ f(u - v) + c\sum_{j=1}^{n} (u_j + v_j) \mid u_j \ge 0, v_j \ge 0 \ \forall j \}.$$

This has nonnegativity constraints. By substituting $u_j = w_j^2$ and $v_j = z_j^2$, we obtain an unconstrained problem:

$$\min_{w,z \in \Re^n} \hat{f}_c(w, z) := f(w_1^2 - z_1^2, \ldots, w_n^2 - z_n^2) + c\sum_{j=1}^{n} (w_j^2 + z_j^2).$$

The objective function $\hat{f}_c$ is C^1 on $\Re^{2n}$. What are the possible drawbacks? (If f is convex, e.g., $f(x) = e^x$ or $f(x) = \|x\|^2$, would $\hat{f}_c$ be convex?)

c) A third way is to adapt our gradient descent method. Given $x^k \in \Re^n$, let d^k be the solution of

$$\min_{d \in \Re^n} (g^k)^T d + \frac{1}{2} d^T H^k d + c\|x^k + d\|_1,$$

where $g^k = \nabla f(x^k)$ and $H^k \in \Re^{n \times n}$ is symmetric positive definite. Find closed form solution for d^k when (i) $c = 0$ and when (ii) H^k is diagonal (so all diagonal entries of H are positive). For (ii), you

may find it convenient to express your answer using the "mid" notation, i.e., $\text{mid}\{a,b,c\} = b$ where $a \le b \le c$. (It can be shown that the method $x^{k+1} = x^k + \alpha^k d^k$, with α^k given by the Armijo rule using the descent condition $f_c(x^k + \alpha d^k) \le f_c(x^k) + \alpha\sigma((g^k)^T d^k + c\|x^k + d^k\|_1 - c\|x^k\|_1)$, achieves global convergence, i.e., every cluster point of $\{x^k\}$ is a stationary point, which is a global minimum whenever f is convex. This seems the best way, especially for large problems.)

#3. [Approximating symmetric matrix by symmetric positive definite matrix.]

a) Let

$$f(x_1, x_2, x_3) = \frac{1}{2}[(x_1)^2 + (x_2)^2 + (x_3)^2] + x_1 x_2 x_3.$$

Let $Q = \nabla^2 f(0,1,2)$ and $g = \nabla f(0,1,2)$. Choose numbers δ_i, $i = 1,2,3$, such that

$$Q + \begin{bmatrix} \delta_1 & 0 & 0 \\ 0 & \delta_2 & 0 \\ 0 & 0 & \delta_3 \end{bmatrix} = LL^T$$

for some lower triangular matrix $L = [l_{ij}]$ with $l_{ii} = 1$ for $i = 1,2,3$. Then, use one forward-solve and one backward-solve to compute the modified Newton direction d satisfying $LL^T d = -g$.

b) Find the Cholesky factorization $Q = LL^T$ for the $n \times n$ positive definite tri-diagonal matrix Q with diagonal entries 2, and -1 along the off-diagonal next to the diagonal. [You can use chol in Matlab to help you.] How many nonzero entries do L, L^{-1} and Q^{-1} have? If n is large (say, $n = 1000$), which of the three would you store and use to solve the linear equation $Qu = -g$? [This equation arises from discretizing the linear PDE $\partial^2 u(x)/\partial x^2 = -g(x)$, $x \in \Re$.]

#4.* Let x^k be the iterate generated by applying the conjugate gradient method $x^{k+1} = x^k + \alpha^k d^k$, $k = 0, 1, ...$, to minimize $f(x) = \frac{1}{2}x^T Q x - b^T x$, where Q is positive definite and symmetric. Let $g^0 = \nabla f(x^0)$. Show, by induction on k, that d^k is a linear combination of $g^0, Qg^0, ..., Q^k g^0$, for $k = 0, 1, ...$

#5. [Computing problem.] Using Matlab or your favorite computer language, implement the CG method (1.169)–(1.171) with β^k given by the Polak-Ribiere formula (1.172). Also, implement the BFGS method (i.e., (1.180), (1.181), (1.185)–(1.187) and $\xi^k = 1$) with $D^0 = I$. Use Armijo rule for both methods. To ensure convergence, put in a steepest descent safeguard as in Method I. (On the class webpage is posted a sample Matlab code that does this.) For BFGS, a safeguard to ensure positive definiteness of D^{k+1} is needed. Run your programs on the Rosenbrock and Powell examples functions from HW 1, and compare with steepest descent method and the Newton method from HW 1. Also run your programs on two high-dimensional example functions from the literature (see "Testing unconstrained optimization software", ACM Trans. Math. Softwr, 7, 1981, page 26):

$$f_3(x) = \left(\sum_{i=1}^n i(x_i - 1)\right)^2 + \left(\sum_{i=1}^n i(x_i - 1)\right)^4 + \sum_{i=1}^n (x_i - 1)^2, \quad x^0 = (\frac{1}{n}, ..., \frac{1}{n})^T, \quad n = 500.$$

$$f_4(x) = \sum_{i=1}^n \left(n - \sum_{j=1}^n \cos x_j + i(1 - \cos x_i) - \sin x_i\right)^2, \quad x^0 = (\frac{1}{n}, ..., \frac{1}{n})^T, \quad n = 1000.$$

Both have global minimum value of zero. [The matlab code for f_4 and its gradient are posted on the class webpage for reference. Matlab, unlike compiler languages like Fortran and C, is inefficient with loops. So, avoid loops possible.] How do your programs compare in terms of (i) number of iterations, (ii) number of function evaluations, (iii) number of gradient evaluations, (iv) CPU time? If function and gradient are expensive to evaluate, which method would you choose? If function and gradient are cheap to evaluate but n is large, which method(s) might you choose? Explain. What do you observe about the stepsizes?